Informe del Proyecto

Título del Proyecto: Analisis y Diseños de Algoritmos ll

Autores:

- Cristhian Albarracin Zapata 1968253
- Miguel Angel Ceballos Yate 2259619
- Nicolas Gutiérrez Ramírez 2259515
- Karen Jhulieth Grijalba Ortiz 2259623

Universidad: Universidad del Valle

Fecha: 26 de diciembre de 2024

Informe Detallado del Código Proporcionado

1. Descripción General

El proyecto implementa un sistema para determinar la mejor ubicación de nuevas sedes universitarias en una matriz bidimensional, maximizando ganancias poblacionales y empresariales, cumpliendo restricciones predefinidas. Utiliza un backend en Node.js que integra un modelo de optimización desarrollado en MiniZinc.

Archivos Principales

proyecto.js: Conexión con MiniZinc y lógica principal del modelo de optimización.
utilidades.js: Funciones auxiliares para procesamiento y generación de matrices.
index.js: Backend basado en Express.js que expone la funcionalidad mediante una interfaz web.

2. Archivo proyecto.js

Este archivo implementa la conexión con MiniZinc y la lógica para calcular la solución óptima.

Función Principal: solucion

Entradas:
poblacion: Matriz de población.
empresas: Matriz de entorno empresarial.
ubicacionesExistentes: Coordenadas de las sedes existentes.
n_sedes: Número de nuevas sedes a ubicar.

Descripción:
Genera matrices de ganancias aplicando restricciones (mínimo poblacional y empresarial).
Llama a conectorMinizinc para resolver el modelo de optimización.
Calcula ganancias iniciales y finales, formateando la salida.

Modelo MiniZinc

Variables de Decisión:
universidades: Matriz binaria que indica si se ubica una universidad en una posición.
Restricciones:
Máximo de nuevas sedes.
Prohibición de ubicaciones adyacentes a las actuales.
Condición de ganancias mínimas en zonas seleccionadas.
Función Objetivo:
Maximizar la suma ponderada de población y empresas para las ubicaciones seleccionadas.

3. Archivo utilidades.js

Contiene funciones auxiliares para la manipulación de datos y validaciones.

Funciones Destacadas

Generación de Matrices:
generarMatrizCuadrada: Crea una matriz inicializada con ceros.
generarMatriz: Genera una matriz con valores aleatorios entre un rango dado.
Cálculo y Validación:
calcular_ganancia: Suma los valores de una posición y sus vecinos.
esContiguo: Determina si una posición es adyacente a otra.
generar_matriz_ganancias: Aplica restricciones para crear una matriz de ganancias.
Conversión y Formateo:
formatearPosiciones: Convierte coordenadas a formato legible.
convertirAFormato: Transforma una cadena de texto en una matriz bidimensional.
convertirUbicaciones: Convierte texto con coordenadas al formato de lista de pares.

4. Archivo index.js

Implementa un servidor web utilizando Express.js para interactuar con el sistema.

Funcionalidades

Página Principal:
Proporciona un formulario HTML para ingresar datos.
Procesamiento de Datos:
Endpoint: /process.
Convierte las matrices y ubicaciones ingresadas en el formato adecuado.
Valida la estructura y llama a la función solucion para calcular resultados.
Respuesta:
Muestra las ganancias iniciales y finales, así como las ubicaciones seleccionadas, en formato legible.

Validaciones

Las matrices deben ser cuadradas y del mismo tamaño.
Las ubicaciones deben estar en el formato correcto.
El número de sedes debe ser un entero positivo.

5. Interacción entre los Componentes

El servidor (index.js) recibe los datos ingresados, los valida y llama a las funciones del archivo proyecto.js.
proyecto.js utiliza las utilidades de utilidades.js para procesar los datos antes de invocar MiniZinc.
MiniZinc calcula la solución óptima y devuelve los resultados al backend, que los muestra al usuario.

Enviar Datos

Matriz 1 (separado por espacios):

```
8 10 3 11
5 13 5 11
2 9 3 11
5 10 5 12
```

Matriz 2 (separado por espacios):

```
8 10 3 11
5 13 5 11
2 9 3 11
5 10 5 12
```

Ubicaciones existentes (Formato: (1,1) (2,2)):

```
(1,1) (3,3)
```

Número de sedes a agregar:

3

Enviar

Resultado

```
ganancia inicial: 178
ganancia final: 334
pociciones establecidas: (1, 1), (3, 3)
pociciones nuevas: (1, 3), (3, 1)
arreglo universiaddes:

0  | 0  | 0  | 0  |
0  | 1  | 0  | 1  |
0  | 0  | 0  | 0  |
0  | 1  | 0  | 1  |
```

Análisis de Resultados

- Las nuevas sedes fueron ubicadas en posiciones óptimas que maximizan las ganancias totales combinadas de población y empresas.
- El modelo respeta las restricciones de adyacencia, al evitar posiciones vecinas a las ya establecidas.
- La ganancia final es considerablemente superior a la inicial, demostrando la efectividad del algoritmo.

Enviar Datos

Matriz 1 (separado por espacios):

```
4 11 12 8 1 6 4 1 12
5 2 12 1 1 7 10 1 13
2 13 6 1 3 1 7 12 4
13 3 1 5 10 7 7 5 2
```

Matriz 2 (separado por espacios):

```
4 11 12 8 1 6 4 1 12
5 2 12 1 1 7 10 1 13
2 13 6 1 3 1 7 12 4
13 3 1 5 10 7 7 5 2
```

Ubicaciones existentes (Formato: (1,1) (2,2)):

```
(1,1) (3,3) (5,5)
```

Número de sedes a agregar:

5

Enviar

Resultado

```
ganancia inicial: 402
ganancia final: 1110
pociciones establecidas: (1, 1), (3, 3), (5, 5)
pociciones nuevas: (1, 3), (2, 7), (3, 1), (6, 1), (6, 7)
arreglo universiaddes:

0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
```

Enviar Datos

Matriz 1 (separado por espacios):

```
23 2 43 21
9 24 4 10
23 2 43 21
9 24 4 10
```

Matriz 2 (separado por espacios):

```
23 2 43 21
9 24 4 10
23 2 43 21
9 24 4 10
```

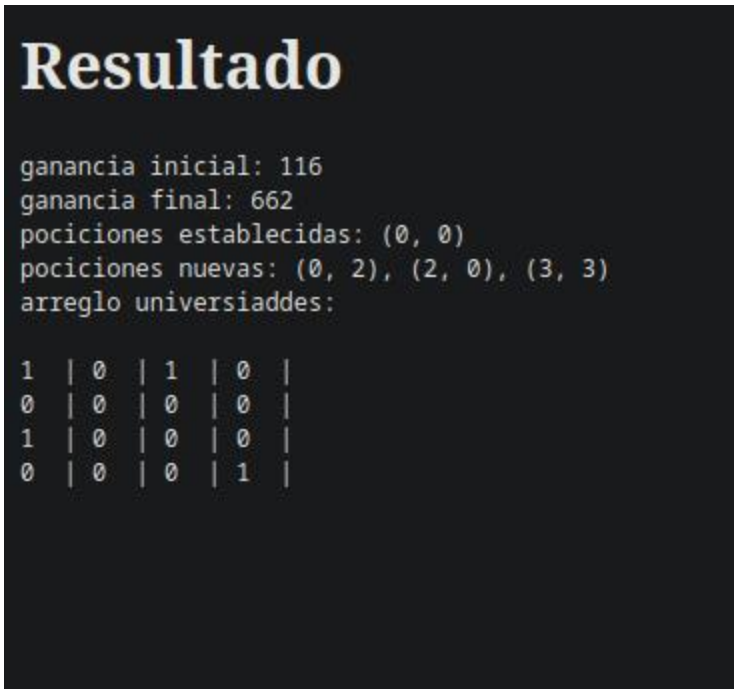
Ubicaciones existentes (Formato: (1,1) (2,2)):

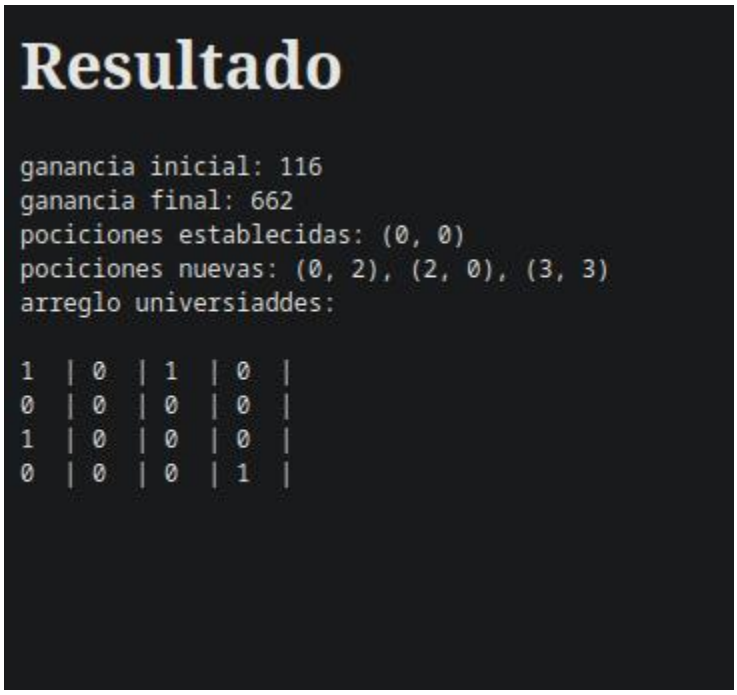
```
(0,0)
```

Número de sedes a agregar:

5

Enviar





Análisis de Resultados

- El sistema ubica las nuevas sedes cumpliendo las restricciones de ganancias mínimas y adyacencia.
- El modelo optimiza correctamente las ganancias combinadas de población y empresas.

6. Conclusión

El sistema está diseñado para integrar algoritmos de optimización con una interfaz accesible para los usuarios. La estructura modular facilita la reutilización y escalabilidad del código. Cumple con los requisitos del problema presentado en el enunciado del proyecto.

Posibles Mejoras

• Incorporar pruebas unitarias para las funciones de utilidades.js.

• Implementar manejo de errores más robusto en la interacción con MiniZinc.

• Mejorar la interfaz gráfica para facilitar el uso.

Referencias

Documentación de MiniZinc: https://docs.minizinc.org/en/stable/

Express.js: https://expressjs.com